Christopher Hernandez
Week 13 Research

1. How do web APIs work? Why are they useful?

Web APIs, or Application Programming Interfaces, enable communication and interaction between different software systems over the internet. They define a set of rules and protocols that allow applications to request and exchange data or perform actions with other applications, services, or platforms.

Here's a high-level explanation of how web APIs work:

1. Request: An application (referred to as the client) initiates a request to a server hosting the API. The request specifies the desired action, such as retrieving data or performing an operation.
2. API Endpoint: The client sends the request to a specific URL called an API endpoint. Each API endpoint represents a specific functionality or resource provided by the API.
3. HTTP Methods: The request typically uses standard HTTP methods like GET, POST, PUT, or DELETE to indicate the type of action the client wants to perform on the resource.
4. Request Parameters: The client may include additional data or parameters in the request, such as filters, search terms, or authentication credentials, depending on the API's requirements.
5. Server Processing: Upon receiving the request, the server hosting the API processes it, executes the requested action, and retrieves the necessary data from databases or other sources.
6. Response: The server generates a response containing the requested data or the result of the requested action. The response is usually in a structured format, such as JSON or XML.
7. HTTP Status Codes: The server includes an HTTP status code in the response to indicate the success or failure of the request. Common status codes include 200 (OK), 400 (Bad Request), 401 (Unauthorized), 404 (Not Found), or 500 (Internal Server Error).
8. Data Exchange: The server sends the response back to the client over the internet.
9. Client Processing: The client receives the response and processes the data or performs further actions based on the information received.

Web APIs are useful for several reasons:

1. Interoperability: APIs enable different software systems, services, or platforms to communicate and interact with each other. They allow applications to leverage functionalities and data from external sources without needing to understand the underlying implementation details.
2. Integration: APIs facilitate the integration of different applications and services, enabling developers to combine functionalities and create new applications or services by leveraging existing ones.

3. Extensibility: APIs allow developers to extend the capabilities of their applications by incorporating features and data from external APIs. This helps in enhancing the functionality and value of applications without having to build everything from scratch.
4. Efficiency: APIs provide a standardized and efficient way to access and exchange data. They enable developers to retrieve specific data or perform actions with minimal effort, as the API handles the underlying complexity.
5. Ecosystem Development: APIs foster the creation of developer communities and ecosystems around platforms or services. They encourage collaboration, innovation, and the development of third-party applications or integrations that extend the reach and usefulness of the platform or service.

For further in-depth understanding of web APIs and their usage, you can explore reputable online resources, API documentation, and tutorials that delve into the specific details and implementation aspects of working with APIs.

2. Look up 3 different public web APIs online and describe 3 endpoints from each.

In your description include the following:

(a) The URI (or endpoint)

(b) All of the HTTP verb methods that can be sent to that endpoint.

(c) A description of what happens when each verb method is sent.

OpenWeatherMap API:

(a) URI/Endpoint: `https://api.openweathermap.org/data/2.5/weather`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to retrieve current weather data for a specific location. You can provide parameters such as city name, geographic coordinates, or ZIP code to get weather information like temperature, humidity, wind speed, etc.

(a) URI/Endpoint: `https://api.openweathermap.org/data/2.5/forecast`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to retrieve weather forecast data for a specific location. You can provide parameters like city name, geographic coordinates, or ZIP code to get forecasted weather information for multiple timestamps in the future, including temperature, humidity, precipitation, etc.

(a) URI/Endpoint: `https://api.openweathermap.org/data/2.5/onecall`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to retrieve weather data for a specific location based on latitude and longitude. This endpoint provides current weather, hourly forecast, and daily forecast information, including temperature, precipitation, UV index, wind speed, and more.

GitHub API:

(a) URI/Endpoint: `https://api.github.com/users/{username}`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to retrieve information about a specific GitHub user. By replacing `{username}` with the actual username, you can fetch details like user profile, repositories, followers, following, public activity, and more.

(a) URI/Endpoint: `https://api.github.com/repos/{owner}/{repo}`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to retrieve information about a specific GitHub repository. By replacing `{owner}` with the repository owner's username and `{repo}` with the repository name, you can obtain details like repository metadata, commit history, contributors, issues, pull requests, and more.

(a) URI/Endpoint: `https://api.github.com/search/repositories`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to search for GitHub repositories based on specific criteria. You can provide parameters like search query, sort order, and filters to get a list of repositories that match the search criteria, along with relevant metadata.

SpaceX API:

(a) URI/Endpoint: `https://api.spacexdata.com/v4/launches`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to retrieve information about SpaceX launches. This endpoint provides details about past, upcoming, and even future launches, including launch dates, mission names, rocket details, launch site information, and more.

(a) URI/Endpoint: `https://api.spacexdata.com/v4/rockets`

(b) HTTP Verb Methods: GET

(c) Description:

- GET: Sends a GET request to fetch information about SpaceX rockets. This endpoint returns data on various rockets used by SpaceX, such as their names, types, dimensions, engine details, payload capacities, and more.

Citations:

https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces
https://www.ibm.com/topics/api
https://en.wikipedia.org/wiki/API
https://openweathermap.org/api
https://docs.spacexdata.com/
https://docs.github.com/en/rest/overview/about-githubs-apis?apiVersion=2022-11-28